Amendments to the Drawings

The attached sheet of drawings includes changes to FIG. 2. This sheet, which includes FIG. 1and FIG. 2, replaces the original sheet including FIG. 1and FIG. 2. In FIG. 2, previously unnumbered element 110 has been numbered and conformed to the new numbering in the specification.

Attachments:

Replacement Sheet

Annotated Replacement Sheet

Remarks/Arguments

Office Action Summary

Status.

1. This *RESPONSE A* is in answer to the Office communication mailed 07/05/2005.
2. The Office communication is non-final.
3. NA

Disposition of Claims.

4. Claims 1 - 21 are pending in the application.
5. No Claims have been allowed.
6. The rejected Claims 1 - 21 have been amended. The claims as presently presented in the application (ORIGINAL, CANCELLED, AMENDED per A) appear as follows under the heading CLEAN CLAIMS AFTER *RESPONSE A*. The claims as presently presented in the application (ORIGINAL, CANCELLED, AMENDED per A) showing changes using a word-processing compare function appear as follows under the heading INTERLINED CLAIMS AFTER *RESPONSE A*.
7. NA
8. NA

Application Papers.

9. The objection to the Specification is believed overcome by amending the Abstract to be shorter.
10. The drawings are objected to and a Replacement sheet has been provided.
11. NA

Priority under 35 U.S.C. § 119.

12. NA

DETAILED ACTION

1 Claims 1-21 as amended by this *RESPONSE A* are presented for reconsideration and examination .

Drawings

2 The drawings have been objected to.

2.1 Regarding FIG 1, Paragraph [0024] of the specification has been amended to conform "executable code 10" as it appears in FIG 1 of the drawings and throughout the specification.

2.2 Regarding FIG 2, "10" has been renumbered as –110–. Similarly, Paragraph [0026] of the specification has been corrected to identify the "group access unit 110". The submitted drawing sheet is labeled "Replacement Sheet" .

Specification

3 The Abstract has been shortened by amendment.

Claim Rejections -35 USC § 101

(1) The Examiner rejected Claims 1-10 under 35 U.S.C. §101 because the claimed invention is allegedly directed to non-statutory subject matter.

(1.1) The citations of the Examiner are noted together with the suggested amendments to overcome the rejection.

(1.2) The independent claims have been amended to recited a "computer-implemented method" and "executing said translated instructions to emulate said legacy instructions" and therefore the rejection under 35 U.S.C. §101 is believed overcome.

Claim Rejections -35 USC § 102

4 The Examiner rejected Claims 1-3, 6-7,11-12 & 17-18 under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter).

4.1 In making the rejection, the Examiner states:

Regrading Claim 1

4.1.1 Mann '295 teaches method for dynamic emulation of target (legacy) instruction by accessing the target (legacy) instruction (Mann '295: Col.2 Lines 44-46, Col.6 Lines 11-12).

4.1.2 Further, Mann '295 teaches for each target instruction is translated to one or more host code blocks (Mann '295: Col.5 Lines 58-62).

4.1.3 Further, Mann '295 teaches setting a flag/tag when an operand setting instruction is encountered, indicating that the value is not present in the translated code'(Mann '295: Col.9 Lines 20).

4.1.4 Further, Mann '295 teaches checking if the flag is set in an operand-using instruction (Mann '295: Col.9 Lines 1-9).

4.1.5 Mann '295 teaches suspending the translation of operand-using instruction, splitting the translated code block into two -up to (before) the point where operand setting instruction is encountered. In next step executing the operand setting instruction and then continuing to translate the remaining block (Mann '295: Col. 7 Lines 14-37;Col.9 Lines 21-36).

4.1.6 Mann '295 teaches translation continues when flag is not set (Mann '295: Col.9 Lines 18-20; Fig. 3 & 5).

4.2 The arguments of the Examiner regarding Claim 1 are traversed for the following reasons.

4.2.1 By way of background, in the present invention, as is clear, for example, from paragraphs [0044] and [0016] (using paragraph numbers of the published application, 20030093776), that

A legacy instruction is translated into one or more translated instructions. If the particular legacy instruction is an operand-setting instruction for storing a value of a precedent operand, a corresponding flag is set when the value of the precedent operand has not been determined. If the particular legacy instruction is an operand-using instruction for using the precedent operand, a check is made to determine if the corresponding flag is set. If the corresponding flag is set, ***translation of the operand-using instruction is suspended and the one or more particular translated instructions corresponding to the operand-setting instruction are executed*** *to determine the value of the precedent operand. Thereafter, the translation of the legacy instructions is resumed using the value of the precedent operand in the resumed translation. If the corresponding flag is not set, the translation of the operand-using instruction continues without suspension. In one specific embodiment, the particular legacy instruction is an unpredictable operand-using instruction having unpredictable byte alignment, for example, because the unpredictable operand-using instruction uses operands of variable length. If the particular legacy instruction is a predictable operand-using instruction, the checking is bypassed and the translation continues, for example, where the predictable operand-using instruction employs fixed-length operands.*

4.2.2 The Examiner relies upon Mann '295 Col 9: lines 1-9 and 9-20 as follows:

FIG. 5 is a flowchart illustrating storing Target system data, in accordance with the present invention. First, the

(Col 9: Line 1)

Target data is stored in Target system memory, step 132. In the preferred embodiment, the store is limited to the number of significant Target system bits. Thus, when emulating the GCOS 8 architecture, 36-bits of data are stored in the Target system memory. A test is then made whether the store was into Target code, step 134. In the preferred embodiment, this test is accomplished by testing the Target code tag 72 associated with the Target word into which the store is being made. A code tag 72 not equal to zero indicates a store into Target code. This in turn indicates self-modifying code. A test is then made whether Host code 88 has already been generated for this Target instruction 76, step 136. If Host code 88 has already been generated for this Target instruction 76, step 136, then the Target instruction is marked with a code tag 72 equal to "X" or "Don't Translate", step 142. This is to suppress subsequent Dynamic Object Code Translation (DOCT) for

> *this Target instruction 76 since that instruction is self-modified code. Then, the block of Host code 88 is either entirely disabled, or is split, step 144. In all cases, the Store functionality then completes, step 148.*

4.2.3 Mann '295, as quoted in Section 4.2.2, sets a tag for self-modifying code that cannot be translated and executed as translated code. Mann '295 aborts both translation and translated code execution and returns execution to the emulator for interpretation execution of the target (legacy) instruction. By way of contrast, the present invention sets a flag to indicate when an operand-setting instruction has not set a precedent operand needed by an operand-using instruction. Rather than aborting translation or translated instruction execution, the present invention suspends the operand-using instruction, executes the translated instructions for setting the precedent operand and then resumes translated execution of the operand-using instruction.

4.2.4 Nowhere in Mann '295 is there any description or suggestion of an operation that would be useful in connection with applicant's invention. For example, if the particular legacy instruction is an unpredictable operand-using instruction, such as one having unpredictable byte alignment because of variable length, the present invention is particularly useful for processing such operands of variable length. Nowhere in Mann '295 is there discussion of unpredictable byte alignment, or more generally, the problem of unpredictable precedent operands. The problem of unpredictable precedent operands is not even addressed in Mann '295. The problem solved by applicant's invention is not even recognized in Mann '295.

4.2.5 Mann '295 addresses the problem of self-modifying code and does not address the problem of unpredictable precedent operands. All of the "operands" in Mann '295 are of fixed length and hence Mann '295 cannot encounter the problem solved by applicant's invention. For purposes of argument and only argument (and with no admission intended), assume that the Examiner's arguments are correct (clearly they are not), then the apparent "solution" provided by Mann '295 could not in fact solve the problem solved by applicant's invention. In the Mann '295 system, the operation is to abort translated code execution and revert to emulation with interpreted code. Such aborting is highly wasteful of execution time and hence does not capture the efficiency of the present invention. The present invention, by merely changing the order of execution of translated instructions (that is, advancing the execution of translated operand-setting instructions), retains the great efficiency of executing only translated instructions without the need to abort to interpretation of the legacy (target) instructions as is done in Mann '295.

4.2.6 The Examiner is apparently using applicant's own specification to find the "teaching " of applicant's invention in Mann '295. However, the logic used by the Examiner is backwards. As is clear from Section 4.2.1, applicant's flag is set when the precedent operand **value has not been determined**. Quite to the contrary, as is clear from Section 4.2.2 above, the "tag" is set in Mann '295 when a **value has been determined** and a store into target code has occurred. The setting of the "flag" in Mann '295 signals that the translation must be aborted and not pursued again in the future. The flag in the present invention signals that the *operand-setting instruction* is to be executed to determine the precedent operand value in advance of the *operand-using instruction* (and no aborting is required). The "tag" in Mann '295 is for an entirely different purpose than the flag of the present invention and Mann '295 suggests nothing about the present invention and particularly about executing translated instructions to determine the precedent operand value in advance of the *operand-using instruction* translated code execution.

4.2.7 As quoted in Section 4.1.3 above, the Examiner concludes "*Further, Mann '295 teaches setting a flag/tag when an operand setting instruction is encountered, indicating that the value is not present in the translated code'(Mann '295: Col.9 Lines 20)*". However, Mann '295 has no discussion of an operand-setting instruction.

4.2.8 As quoted in Section 4.1.4 above, the Examiner concludes, "*Further, Mann '295 teaches checking if the flag is set in an operand-using instruction (Mann '295: Col.9 Lines 1-9)*" However, Mann '295 has no discussion of an operand-using instruction.

4.2.9 As quoted in Section 4.1.5 above, the Examiner concludes, *Mann '295 teaches suspending the translation of operand-using instruction, splitting the translated code block into two -up to (before) the point where operand setting instruction is encountered. In next step executing the operand setting instruction and then continuing to translate the remaining block (Mann '295: Col.7 Lines 14-37;Col.9 Lines 21-36).* However, Mann '295 has no discussion of an operand-using or operand-setting instructions. Mann '295 does not have an operation where ***translation of the operand-using instruction is suspended and the one or more particular translated instructions corresponding to the operand-setting instruction are executed.*** Mann '295 aborts execution of the translated code and returns execution to the emulator for interpreted operation. Mann '295 does split the code into two parts, that is the part that cannot be translated and the part that can be translated. Mann '295 does not suspend and resume as in the present invention. Mann '295 aborts, interprets and only then resumes translation.

4.3 The Examiners characterization of Mann '295 for Claims 2 & 3 is traversed as follows:

4.3.1 The Examiner states,

Regarding Claim 2 & 3

Mann '295 anticipates same functionality as "resume-translation call" by splitting the code block into two blocks, allowing the execution of the first block (by host) as translated before and interpreting the second block, so that the operand using instruction is translated after the operand setting instruction is executed (Mann '295: Col.9 Lines 37-52; Col.7 Lines 4-12). Hence the resume-translation call is inherent in the design.

4.3.2 Mann '295 does not resume the suspended operation by inserting "*a resume_translation call into a stream of said translated instructions*". Rather, Mann '295 aborts the stream of translated instructions and executes by interpretation.

4.4 The Examiners characterization of Mann '295 for Claims 6 & 7 is traversed as follows:

4.4.1 The Examiner states,

Regarding Claim 6 & 7

Mann '295 teaches bypassing the checking if there is no store instruction data (Mann '295: Fig. 5 Element 134/136 & 148). Further, Mann '295 teaches operand-using instructions employing fixed length operands (Mann '295: Fig. 3 Element 76).

4.4.2 In Section 4.1.4 above, the Examiner identifies the "checking" as *"Further, Mann '295 teaches checking if the flag is set in an operand-using instruction (Mann '295: Col.9 Lines 1-9)*" so that according to the Examiner's interpretation, the "checking" is always performed and no bypassing occurs. The checking in Col. 9 Lines 1-9 identified by the Examiner is element 134 in FIG 5 of Mann '295 and this checking is not bypassed.

4.5 The Examiners characterization of Mann '295 for Claims 12-14, 17 and 18 is traversed as follows:

4.5.1 The Examiner states that these rejections are the same as for prior claims, for example, regarding Claim 11, the Examiner states:

Regarding Claim 11

Claim 11 discloses same limitations as claim 1 and is rejected for the same reasons as claim 1.

4.6 The Examiners rejection based upon Mann '295 for Claims 12-14, 17 and 18 is traversed for the same reasons as for Claim 1 and the other claims as discussed above.

Claim Rejections -35 USC § 103

5 The Examiner rejected Claims 4-5,8-10, 15-16 & 19-21 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,516,295 issued to George A. Mann et al (Mann '295 hereafter) in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).

5.1 In making the rejection, the Examiner states Regarding Claim 4, 5 & 8:

Regarding Claim 4, 5 & 8

Teaching of Mann '295 is disclosed in claim 1 rejection above. Mann '295 does not teach unpredictable operand using instructions like MVC instruction for IBM S/390 embodiment disclosed in the specification [0031].

Scalzi '013 teaches an instruction set emulation of S/390 (CISC based processor) on Power PC (RISC based processor), similar to the one disclosed by the applicant. Since the MVC-like operand-using instructions for IBM S/390 were in the art at the time Scalzi '013 teachings were provided; translation of all legacy instructions from IBM S/390 to RISC based processor would have been obvious and necessitated in Scalzi '013 design.

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to apply teachings of Scalzi '013 to Mann '295 to translate legacy operand-using legacy instructions containing variable length operands with different byte alignments. The motivation to combine would have been that Mann '295 and Scalzi '013 are analogous arts solving the problem of dynamic binary translation from CISC instruction set architecture to RISC-based architecture (Scalzi '013: Col.17 Lines 54-57; Mann '295: Col.3 Lines 62-67).

Regarding Claim 9

Both, Scalzi '013 & Mann '295 teach, that the legacy instructions are object code instructions compiled and assembled for legacy architecture (Scalzi '013: Col.4 Lines 37-46; Mann '295: Col.2 Lines 44-51).

Regarding Claim 10

Scalzi '013 teaches that the translated instructions are for execution in a Power PC architecture, which is a RI8C-based architecture (8calzi '013: Col.17 Lines 54-57).

Regarding Claims 15 & 16

Claims 15 & 16 disclose same limitations as claim 4 & 5 respectively and are rejected for the same reasons as claim 4 & 5.

Regarding Claim 19

Claim 19 discloses same limitations as claim 8 and is rejected for the same reasons as claim 8.

Regarding Claim 20

Claim 20 discloses same limitations as claim 9 and is rejected for the same reasons as claim 9.

Regarding Claim 21

Claim 21 discloses same limitations as claim 10 and is rejected for the same reasons as claim 10.

5.2 The arguments of the Examiner regarding Claims 4-5,8-10, 15-16 & 19-21 are not believed to be well founded. As far as applicant can determine, nothing in either Mann '295 or Scalzi '013 discuses (i) execution for unpredictable operands such as *variable length operands with different byte alignments*, (ii) how such operands are translated to translated instructions and (iii) how the translated instructions are executed. Since neither reference has any such discussion or "teaching", clearly the combination has no teaching either. The Examiner's rejection is traversed for all of the reasons set forth in connection with Claim 1 and other claims above.

6 Reconsideration of all Claims 1 -21, as amended is requested.

Respectfully submitted,

SIGNATURE OF ATTORNEY		
David E. Lovejoy (Reg. No. 22,748)	David E. Lovejoy David E. Lovejoy	**Signature Date:** 4 January 2006
Address	Customer No.	Communication
102 Reed Ranch Rd. Tiburon, CA 94920-2025	21603	Tel: (415) 435-8203 Fax: (415) 435-8857 e-mail: david.lovejoy@sbcglobal.net

FIG. 1

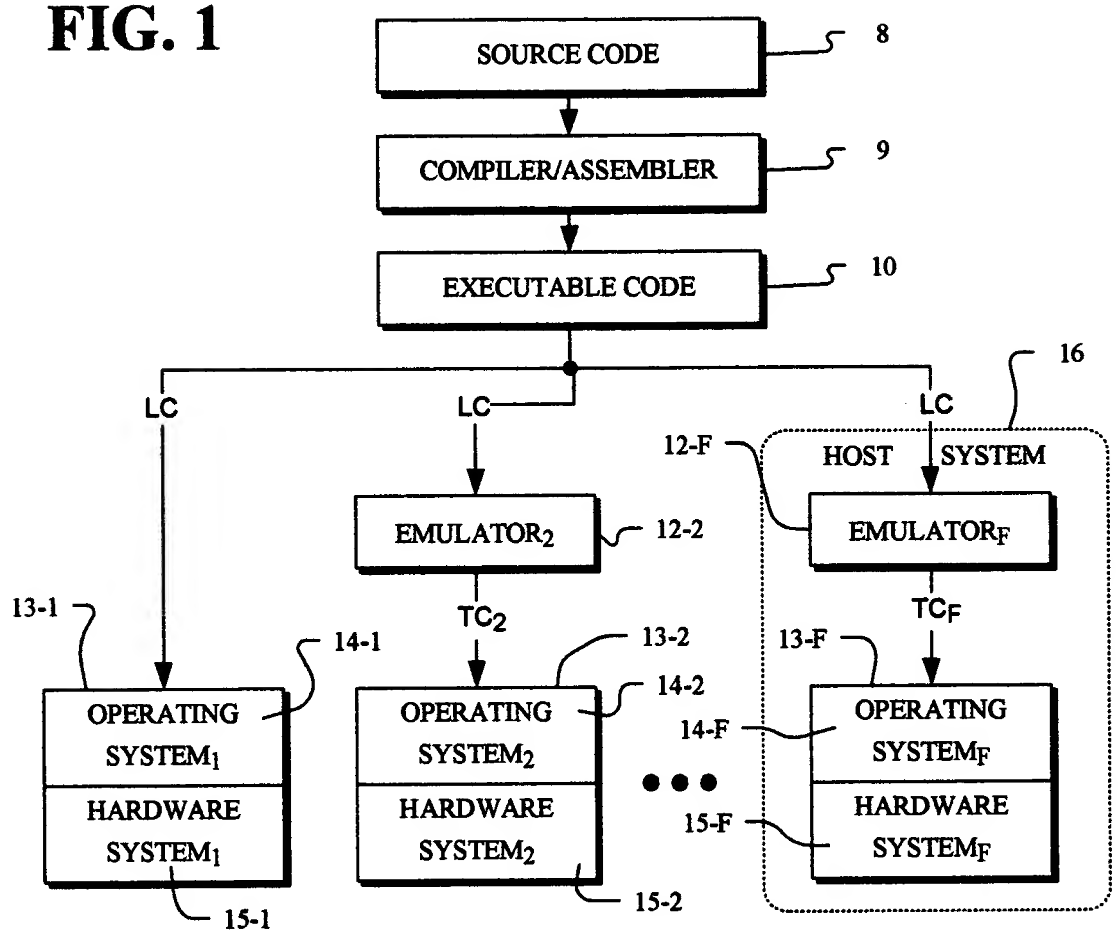
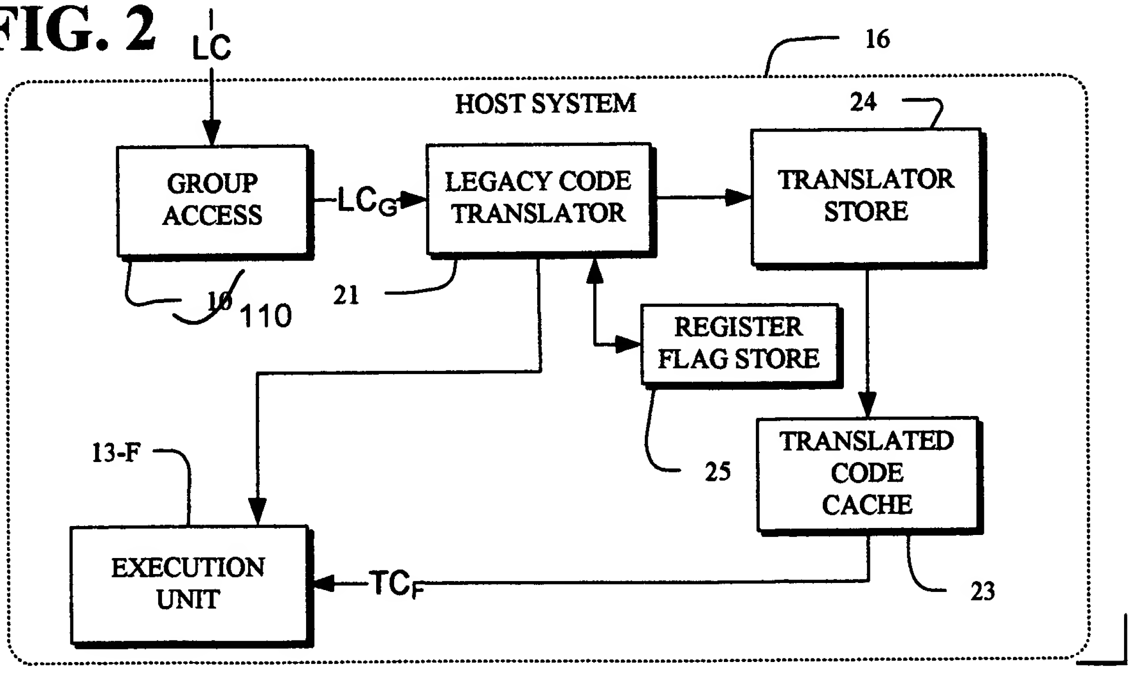

SOURCE CODE
8
COMPILER/ASSEMBLER
9
EXECUTABLE CODE
10
LC
LC
LC
16
12-F
HOST SYSTEM
EMULATOR$_2$
12-2
EMULATOR$_F$
TC$_2$
TC$_F$
13-1
14-1
13-2
13-F
14-2
14-F
OPERATING SYSTEM$_1$
HARDWARE SYSTEM$_1$
OPERATING SYSTEM$_2$
HARDWARE SYSTEM$_2$
OPERATING SYSTEM$_F$
HARDWARE SYSTEM$_F$
15-F
15-1
15-2

FIG. 2


LC
16
HOST SYSTEM
24
GROUP ACCESS
LC$_G$
LEGACY CODE TRANSLATOR
TRANSLATOR STORE
~~16~~ 110
21
REGISTER FLAG STORE
25
TRANSLATED CODE CACHE
13-F
EXECUTION UNIT
TC$_F$
23